John von Neumann Institute for Computing

A Comparative Evaluation of Two Techniques for Predicting the Performance of Dynamic Enterprise Systems

D.A. Bacigalupo, S.A. Jarvis, L. He, D.P. Spooner,
D. Pelych, G.R. Nudd

published in

Parallel Computing:
Current & Future Issues of High-End Computing,
Proceedings of the International Conference ParCo 2005,
G.R. Joubert, W.E. Nagel, F.J. Peters, O. Plata, P. Tirado, E. Zapata (Editors),
John von Neumann Institute for Computing, Jülich,
NIC Series, Vol. **33**, ISBN 3-00-017352-8, pp. 163-170, 2006.

http://www.fz-juelich.de/nic-series/volume33

A Comparative Evaluation of Two Techniques for Predicting the Performance of Dynamic Enterprise Systems

David A. Bacigalupo [a], Stephen A. Jarvis[a], Ligang He[a], Daniel P. Spooner[a], Denis Pelych[b] and Graham R. Nudd[a]
daveb@dcs.warwick.ac.uk

[a]Department of Computer Science, University of Warwick, Coventry CV4 7AL, UK

[b]The National B2B Centre, Coventry CV4 7AL, UK

This paper is concerned with predicting the response times an enterprise information system would provide on new server architectures. These predictions can allow a workload to be transferred onto new servers whilst maintaining quality of service levels. Two common techniques are solving queuing models and extrapolating from previously gathered performance data. The dynamic recalibration of a layered queuing model and a historical model are investigated experimentally using an established distributed enterprise benchmark. The conclusions provide guidelines as to how to select an appropriate technique, including how to dynamically calibrate each model at a low overhead. Using these guidelines it is shown that both techniques can make low overhead predictions for new server architectures at a good level of predictive accuracy (typically over 80%) [1].

1. Introduction

It has been shown that response time predictions can enhance the workload and resource management of enterprise information systems [1,2]. Two common approaches used in the literature for making these response time predictions are extrapolating from historical performance data and solving queuing network models. Examples of the first approach include the use of both coarse [3] and fine [1] grained historical performance data. The former involves recording workload information and operating system/database load metrics, and the later involves recording the historical usage of each machine's CPU, memory and IO resources by different classes of workload. Another example of this approach is being developed in the High Performance Systems Group at the University of Warwick [4]. This historical technique has been implemented as a tool called HYDRA which has been applied to both distributed enterprise [4,5] and business-to-business [6] applications. It is differentiated from other historical modelling work by its focus on simplifying the process of analysing any historical data so as to extract the small number of trends that will be most useful to a resource management system.

Examples of the queuing modelling approach include [7,8,2] and the layered queuing technique, as implemented in the layered queuing network solver (LQNS) [9]. The layered queuing technique is of particular interest and will be examined further in this paper as: it explicitly models the tiers of servers found in this class of application, and it has been applied to a range of distributed systems (i.e. [10]) including the distributed enterprise benchmark used in this paper [11].

It is important to compare the effectiveness of different approaches for modelling enterprise information systems so practitioners can make an informed choice when designing prediction-enhanced workload and resource management systems. However, although there have been comparisons of

[1]The work is sponsored in part by the EPSRC (contract no. GR/S03058/01 and GR/R47424/01), the NASA AMES Research Center administered by USARDSG (contract no. N68171-01-C-9012) and IBM UK Ltd.

different performance prediction approaches using enterprise information systems, there have been few quantitative comparisons of the two approaches on a single enterprise application. For example in [10] a layered queuing model of a distributed database system is created and compared to a markov chain-based queuing model of the system. In [9] the layered queuing technique is compared more generally to other performance modelling techniques. Another recognised queuing technique which has been applied to similar applications is described in [7] and compared with the layered queuing technique. However none of these papers include a comparison with a historical model of the same application. The historical prediction technique described in [3] is applied to a web-based stock trading application (Microsoft FMStocks) and compared to a queuing modelling approach. However a queuing network model is not created of the application.

This paper describes such a comparison for dynamic enterprise information systems. These are systems which must continually adapt to changes in the workload, system configuration (i.e. monitoring and logging policies) and available servers. This may involve workload managers acquiring new servers for which only a small number of benchmarks have been run (i.e. to determine request processing speed). We investigate how the prediction models can be rapidly recalibrated with low overheads on an established server, whilst still obtaining enough data to make accurate predictions on new server architectures. This allows the models to be recalibrated prior to making real-time workload management decisions, and removes the need to model the workload and system configuration variables that change less frequently at runtime. This has a number of advantages when predicting the performance of dynamic systems including: i.) a reduction in model complexity which can dramatically improve the responsiveness of predictions; and ii.) removing the need to consider some variables which may be complex to measure and model (such as the complexity of processing the data in the database for each service class in the workload).

The comparison involves comparing the HYDRA historical technique and the layered queuing technique using a distributed enterprise application benchmark. The contributions of the work are to: i.) investigate the dynamic recalibration of these models experimentally; ii.) provide guidelines for making accurate predictions using both techniques at a low overhead; and iii.) comparatively evaluate the two techniques to help users decide which technique to use. The IBM Websphere middleware [12] is selected as the platform on which the benchmark will be run as it is a common choice for distributed enterprise applications. The IBM Performance Benchmark Sample 'Trade' [13] is selected as it is the main distributed enterprise application benchmark for the Websphere platform.

This remainder of this paper is structured as follows: defining a system model, sample workload and experimental setup (see section 2); investigating the recalibration overheads and accuracies of the two techniques (see sections 3 and 4); and describing the recalibration guidelines and comparative evaluation (see section 5).

2. System Model and Sample Workload

Based on established work (i.e. [11,14]) the enterprise information system is modelled as a tier of application servers accessing a single database server. Application servers may have heterogeneous server architectures. Based on the queuing network in the Websphere e-Business platform: a single first in first out (FIFO) waiting queue is used by each application server; the database server has one FIFO queue per application server; and both types of server can process multiple requests concurrently via time-sharing. The workload consists of clients (divided into service classes) which send requests to the system. The workload manager adjusts the routing of the incoming requests to the application servers which may involve acquiring new servers from another system or from a

computational grid.

The sample workload is as follows. A service class is created for 'browse' users with the next operation (i.e. buy/sell/quote etc) called by a client being randomly selected, with probabilities defined as part of the Trade benchmark. For simplicity, the typical workload is defined as all browse clients. A service class is created for 'buy' users which involves clients making an average of 10 buy requests. 'No. of clients and the mean client think-time' is used as the primary measure of the workload from a service class. Using number of clients (as opposed to a static arrival rate definition) to represent the amount of workload is common when modelling enterprise information systems (i.e. [8,11]). This is because it explicitly models the fact that the time a request from a client arrives is not independent of the response times of previous requests, so as the load increases the rate at which clients send requests decreases. Think-times are exponentially distributed with a mean of 7 seconds for all service classes as recommended by IBM for Trade clients, although heterogeneous think-times are supported by both techniques.

The experimental setup contains 3 application servers running Websphere Application Server (v4.0.1). Under the typical workload the max throughputs of the new 'slow' server $AppServ_S$ (P3 450Mhz), the established 'fast' server $AppServ_F$ (P4 1.8Ghz) and the established 'very fast' server $AppServ_{VF}$ (P4 2.66Ghz) are found to be 86, 186 and 320 requests/second respectively under the typical workload. The database is DB2 7.2 (on an Athlon 1.4Ghz) and 250 clients are simulated by each workload generator (P4 1.8Ghz). All servers run Windows 2000, have at least 512MB RAM and are connected via a 100Mbps switch.

3. The Layered Queuing Technique

A layered queuing performance model explicitly defines an application's queuing network. An approximate solution to the model can then be generated automatically, using the layered queuing network solver (LQNS). The solution strategy involves dividing the queues into layers corresponding to the tiers of servers in the system model, generating an initial solution and then iterating backwards and forwards through the layers solving the queues in each layer by mean value analysis and propagating the result to the next layer until the solutions converge. Performance metrics generated include response times, throughputs and utilisation information for each service class. A detailed description of the layered queuing technique can be found in [9].

A layered queuing model is created with application server, database server and database server disk layers, each layer containing a queue and a processor. The application server disk is not modelled as the Trade applications utilisation of this resource is found to be almost 0 during normal operation. Workload parameters (per service class) are: the number of clients, the mean processing times on each processor, and the average number of database requests per application server request. Processing times are assumed to be exponentially distributed. Queue parameters include the maximum number of requests each processor can process at the same time via time-sharing. Communication time is represented as a constant delay which is calibrated by subtracting the predicted response time from the actual response time at a small number of clients (250 in the experimental setup) on an established server. In the experimental setup the application server, database server and database disk can process 50, 20 and 1 requests at the same time, respectively. And it is found that the buy service class makes 2 database requests, and the browse service class makes 1.14 database requests on average.

The service class mean processing times are calculated during recalibration by taking an established server off-line and sending a workload consisting only of that service class. This overcomes the difficulties that have been found measuring mean processing times (without queuing delay) of

multiple service classes, in real system environments [14]. The mean processing time is then calculated by dividing the CPU/disk usage for the server/database disk, respectively by the throughput (in requests/second). Calculating the service class mean processing times on a new server architecture involves multiplying the mean processing times on the established server by the established/new server request processing speed ratio.

Experiments are conducted to examine the predictive accuracy and resource usage overhead when calibrating the request processing times under different amounts of background workload. The typical workload is calibrated on an established server with a maximum throughput of 213 requests/second. Each test run involves activating the required number of clients and waiting 1 minute for the system to reach a steady state. The %CPU/disk usage samples are then recorded for a period of 1 minute along with the mean throughput of the servers during the minute. The sampling interval is set at 6 seconds so the increase in the %CPU/disk usage is no more than 5%. Predictive accuracy is defined as:

$$accuracy = \frac{|predicted\_value - measured\_value|}{measured\_value} \times 100 \quad (1)$$

It is found that as the number of clients is increased the mean processing time does not remain constant. Instead it decreases (from 5.6ms at 63 clients) to a minimum of 4.3ms at 750 clients and 45% application server CPU usage, and then increases (to 4.7ms at 2250 clients). This pattern is explained as follows. The higher mean processing times at smaller number of clients are due to the larger system and JVM overhead (i.e. for garbage collection) per request. The higher mean processing times at larger numbers of clients are due to the overhead of running a larger number of threads (as Websphere terminates threads that are not needed, at lighter loads). At intermediate numbers of clients these overheads are less significant resulting in lower mean processing times.

As a result of this variation in mean processing time the predictive accuracy is highest when calibrating the model at a number of clients between the maximum and minimum mean processing times. This can be observed in figure 1 which shows the accuracy of predictions on the established server when calibrating the model at different numbers of clients. However the higher the number of clients used for the recalibration the more server capacity that must be taken offline to run the workload generators, application server and database server. In this case the maximum predictive accuracy when the model is recalibrated at a low overhead is at 125 clients. When the number of clients used for the calibration is reduced below 125 the predictive accuracy drops significantly. This is due to a discontinuity in the rate at which the mean response time increases with number of clients around the point at which max throughput is reached. The accuracy drops because the accuracy sample at this point is very inaccurate due to the point at which this discontinuity occurs being predicted incorrectly. Due to this discontinuity the predictive accuracy at the first maximum is slightly lower than that of the second maximum.

When the model is calibrated at 125 clients the mean processing times for the typical workload are measured as 4.675ms, 1.821ms and 0.638ms for the application server, database server and database server disk respectively. This results in a predictive accuracy of 84% on the new ($AppServ_S$) server architecture – see figure 2. A server capacity of 89 requests/second must be taken offline for this calibration for 2 minutes per service class. This is the equivalent of a Pentium III 450Mhz which is likely to be a low recalibration overhead for a modern resource management system.

4. The Historical Technique

The historical modelling technique involves sampling performance metrics (i.e. response times) and associating these measurements with variables representing the workload being processed and

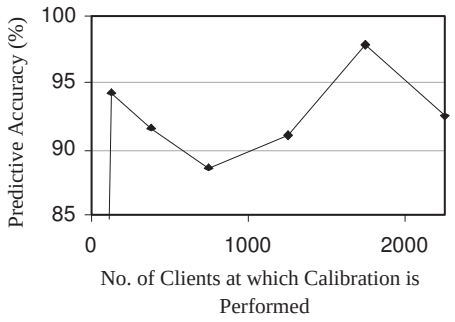

Figure 1. The predictive accuracy when recalibrating the layered queuing model at different loads

Server	CL (ms)	Lambda L (ms)
S	138.9	4E-06
F	84.1	0.0001
VF	10.7	0.0009

Table 1. Historical technique relationship parameters

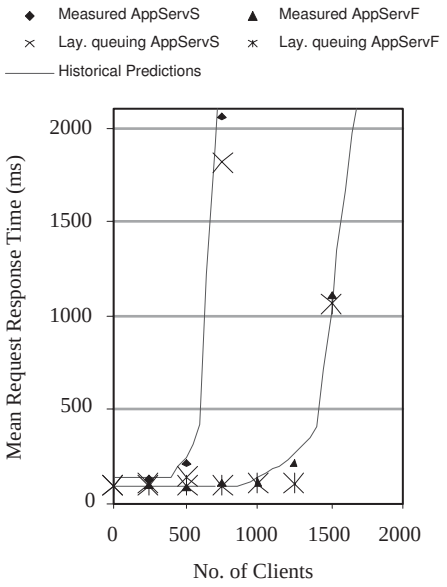

Figure 2. Performance predictions for new (AppServ S) and established (AppServ F) server architectures

the machines architecture. Historical models then define the relationships (i.e. linear/exponential equations) between the variables and metrics. In the case study the server workload variables are the number of clients and the percentage of buy requests, and the main server architecture variable is request processing speed. This results in three relationships.

Relationship 1 models the effect of the number of typical workload clients on the mean response time. It has been found that this relationship is best approximated using separate 'lower' and 'upper' equations for before and after max throughput:

$$mrt_L = c_L e^{(\lambda_L \times no\_of\_clients)} \tag{2}$$

$$mrt_U = \lambda_U \times no\_of\_clients + c_U \tag{3}$$

where mrt_L/mrt_U is the mean response time before/after max throughput respectively, and c_L, c_U, λ_L and λ_U are parameters that must be calibrated from historical data. The correct choice of the lower or the upper equation is made using a relationship between the number of clients and the servers throughput to calculate the number of clients at max throughput. It is also found that using a further breakdown of the possible system loads, so as to define a 'transition' relationship for phasing from the lower to the upper equation, can increase predictive accuracy as discussed in [5]. However the accuracy of such a relationship is not considered further here.

Relationship 2 models the effect of application server max throughput on relationship 1 as follows:

$$c_L = \Lambda(c_L) \times mx\_throughput + C(c_L) \tag{4}$$

$$\lambda_L = C(\Lambda_L) \times mx\_throughput^{\Lambda(\lambda_L)} \tag{5}$$

where $\Lambda(c_L)$, $C(c_L)$, $C(\lambda_L)$ and $\Lambda(\lambda_L)$ are parameters that must be calibrated from historical data. Parameters for the upper (linear) equations are calculated as follows. Given an increase/decrease in server max throughput of z%, λ_U is found to increase/decrease by roughly $1/z$%, and c_U is found to be roughly constant.

Relationship 3 models the effect of the percentage of buy requests in the workload on the servers max throughput. There is found to be a linear relationship between the percentage of buy requests on an established server and its max throughput. This is used to extrapolate $mx\_thr_E(b)$, the max throughput of an established server under a percentage of buy requests, b. The max throughput on a new server at a particular percentage of buy requests, $mx\_thr_N(b)$, is then calculated as follows, where a percentage of buy requests of 0 represents the typical workload:

$$mx\_thr_N(b) = \frac{mx\_thr_E(b)}{mx\_thr_E(0)} \times mx\_thr_N(0) \quad (6)$$

The remainder of this section investigates the calibration of historical models on a live system. This allows a historical model to be calibrated at a significantly smaller resource usage overhead than the layered queuing method as the only additional requirement on the system is to process the one (or more) response time sampling clients. The parameters in relationships 1 and 2 are calibrated by fitting least squares trend-lines to historical data from the established $AppServ_F$ and $AppServ_{VF}$ servers. The historical data consists of the max throughputs of each server and n_{udp}/n_{ldp} data points for the upper/lower equation of relationship 1 respectively. Each data point records the mean response time (averaged across n_s samples) of the typical workload at a numbers of clients.

The overall predictive accuracy is defined as the mean of the lower equation accuracy and the upper equation accuracy. It is found that accurate predictions can be made even when n_{udp} and n_{ldp} are both reduced to 2 and n_s is reduced to 50. The resulting parameters are shown in table 1. Figure 2 illustrates the mean response time predictions made using this calibration (including a transition exponential relationship for phasing between equations 2 and 3). A minimum of 100 samples per 'measured' data point are recorded. A good level of accuracy of 89% for the established servers and 83% for the new server is achieved. For these predictions the 50 samples for each data point were made sequentially (after a 1 minute warm-up period) using only one sampling client. This said, the calibration was completed in only 2 minutes. Relationship 3 can also be rapidly calibrated as this only requires one additional item of data; the max throughput of an established server under a heterogeneous workload. This is tested using LQNS predictions for historical data; specifically the max throughput of $AppServ_F$ under 25% buy requests (158 requests/second). The resulting predictive accuracy is 74% on the new server architecture.

5. Guidelines and Comparative Evaluation

It has been shown that both techniques can make accurate predictions at a low calibration overhead. Our guidelines for achieving these high levels of predictive accuracy using the layered queuing method are as follows based on the experimental analysis in section 3. The key variable is the number of clients at which the layered queuing model is calibrated. This is because the layered queuing technique assumes that the per-service class request processing times are constant at different server loads. However this was not found to be the case due to system (i.e. garbage collection) and thread overheads at small and large numbers of clients, respectively. As this variable, and hence the server load, is increased so does the calibration overhead. We have found that there tends to be a minimum number of clients that gives a high accuracy and low overhead (125 clients in our setup - see figure 1). The procedure in section 3 should be used to identify this point. Alternatively if spare machines

are available to calibrate the model the procedure can be used to locate the high overhead calibration point that gives the maximum accuracy (see figure 1). In our experimental setup this point was at 1750 clients.

Our guidelines for achieving accurate predictions at a low overhead using the historical method are as follows based on the experimental analysis in section 4. It is necessary to sample the response times of established servers for both before and after max throughput is reached due to the discontinuity in the response time scalability graph at this point (see figure 2). It is also necessary to use two (or more) established application server architectures for calibration so as to be able to extrapolate a trend-line (unlike the layered queuing technique which only requires one). Further, when sampling the response times of an established server for either before or after max throughput it is necessary to include samples at both low and high numbers of clients so as to get a sufficient spread of data points from which to draw a trend-line. It has also been found that the predictive accuracy before max throughput is reached tends to be less than the predictive accuracy after max throughput is reached. This is due to the predictions being made using exponential and linear relationships respectively. Initial experiments have shown that exponential predictive accuracy increases to linear predictive accuracy levels if three or more (no. of clients, mean response time) data points are used for calibration as opposed to the current two. We therefore recommend that a minimum of two/three data points (with at least 50 samples per data point) be used when calibrating linear/exponential trend-lines, although in practice the more data points used the better.

When selecting which performance prediction technique to use we recommend that the following criteria be considered: recalibration requirements; the responsiveness of predictions; the systems which can be modelled, the metrics which can be predicted, the ease of use of each technique and the performance modelling expertise required. Model recalibration has been considered above and in the previous two sections; the following is a summary of how the techniques differ in the other categories. For a more detailed discussion the reader is referred to [4].

There are a number of functional limitations with the layered queuing technique that should be taken into account when selecting a technique. It requires significant CPU time to make the mean response time predictions (up to 3 seconds on an Athlon 1.4Ghz under a convergence criterion of 20ms in these experiments), whereas the historical predictions are almost instantaneous. It is also more difficult to model applications that cache significant amount of database data at the application server (as opposed to applications such as the Trade benchmark which access the majority of database data directly so as to avoid data inconsistencies if the application server crashes). This is because the number of calls to the database must be a constant in the layered queuing model, whereas if a cache is used this value will depend on the cache miss rate. Using the historical method the size of the application servers cache can be recorded as an extra variable. Relationships can then be added to approximate the historical relationship between the performance metrics, this new variable and the existing variables, using the techniques presented in section 4. Another limitation of the layered queuing technique is that the important class of percentile response time metrics cannot be predicted directly. In contrast the historical technique can extrapolate from and hence make predictions for a wide range of metrics. However layered queuing models are also significantly easier to create with a minimum level of performance modelling expertise than a historical model. This is because creating a historical model involves specifying and validating how predictions will be made, whereas once a system's queuing network configuration is specified layered queuing models can be solved automatically. The layered queuing technique may therefore be preferable when there is a shortage of either time or performance modelling expertise when creating the model.

6. Conclusion

This paper comparatively evaluates the layered queuing and historical techniques for predicting response times of dynamic enterprise information systems on new server architectures. Based on detailed experimental analysis we provide guidelines for selecting a technique and obtaining low overhead high accuracy predictions. The combination of an established system model, a popular middleware (IBM Websphere) and a distributed enterprise benchmark based on best practices (the Websphere Performance Benchmark Sample) should make this work of relevance to a wide range of enterprise information systems. This is also, to the best of our knowledge, the only quantitative comparison of these two classes of prediction technique on this benchmark. Future work includes evaluating the strengths and weaknesses identified with each technique on different types of prediction-enhanced workload management algorithm.

References

[1] J. Aman, C. Eilert, D. Emmes, P. Yocom, D. Dillenberger: Adaptive Algorithms for Managing a Distributed Data Processing Workload. IBM Systems Journal. 36(2), 1997, 242-283.

[2] Z. Liu, M.S. Squillante, J. Wolf, On Maximizing Service-Level-Agreement Profits, Proc. ACM Conference on Electronic Commerce (EC01), Florida, USA, October 2001

[3] M. Goldszmidt, D. Palma, B. Sabata, On the Quantification of e-Business Capacity, Proc. ACM Conference on Electronic Commerce (EC01), Florida, USA, October 2001

[4] D.A. Bacigalupo, S.A. Jarvis, L. He, D.P. Spooner, D.N. Dillenberger, G.R. Nudd, An Investigation into the Application of Different Performance Prediction Methods to Distributed Enterprise Applications, The Journal of Supercomputing, 34:93-111, 2005

[5] D.A. Bacigalupo, S.A. Jarvis, L. He, G.R. Nudd, An Investigation into the Application of Different Performance Prediction Techniques to e-Commerce Applications, Workshop on Performance Modelling, Evaluation and Optimization of Parallel and Distributed Systems, Proc. 18th IEEE Int. Parallel and Distributed Processing Symposium (IPDPS04), New Mexico, USA, April 2004

[6] J.D. Turner, D.A. Bacigalupo, S.A. Jarvis, D.N. Dillenberger, G.R. Nudd, Application Response Measurement of Distributed Web Services, Int. J. of Computer Resource Measurement, 108, 2002, 45-55

[7] D. Menasce, Two-Level Iterative Queuing Modeling of Software Contention, Proc. 10th IEEE Int. Symposium on Modeling, Analysis and Simulation of Computer and Telecommunications Systems (MASCOTS02), Texas, USA, October 2002

[8] Y. Diao, J. Hellerstein, S. Parekh, Stochastic Modeling of Lotus Notes with a Queueing Model, Proc. Computer Measurement Group Int. Conference (CMG01), California, USA, December 2001

[9] C.M. Woodside, J.E. Neilson, D.C. Petriu, S. Majumdar, The Stochastic Rendezvous Network Model for Performance of Synchronous Client-Server-like Distributed Software, IEEE Trans. On Computer, 44(1), 1995, 20-34

[10] F. Sheikh, M. Woodside, Layered Analytic Performance Modelling of a Distributed Database System, Proc. Int. Conference on Distributed Computing Systems (ICDCS97), Maryland USA, May 1997

[11] T. Liu, S. Kumaran, J. Chung, Performance Modeling of EJBs, Proc. 7th World Multiconference on Systemics, Cybernetics and Informatics (SCI03), Florida USA, 2003

[12] M. Endrel, IBM WebSphere V4.0 Advanced Edition Handbook, IBM Int. Technical Support Organisation Pub., 2002. Available at: http://www.redbooks.ibm.com/

[13] IBM Websphere Performance Sample: Trade. Available at www.ibm.com/software/info/websphere/

[14] L. Zhang, C. Xia, M. Squillante, W. Nathaniel Mills III, Workload Service Requirements Analysis: A Queueing Network Optimization Approach, Proc. 10th IEEE Int. Symposium on Modeling, Analysis, & Simulation of Computer & Telecommunications Systems, Texas, USA, October 2002